United International **University (UIU)**

Dept. of Computer Science & Engineering (CSE)

Final Exam :: Summer 2021

Course Code: CSE 1115 **Course Title: Object Oriented Programming**

Total Marks: **25** Time: **1hr 15 min**

READ THIS CAREFULLY: Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules

Question 1 (2 + 3) [CO1]

Consider the following code:

```
public class DemoClass{
    private double PI = 3.1416;
    void calculateArea(double radius){
        double area = PI*radius*radius;
        System.out.println("Area: " + area);
    }
    public static void main(String []args){
        calculateArea(5.0);
    }
}
```

a) Suggest modification to the code so that the value of **PI** cannot be modified later. Also modify the above code so that it can run without any error.

b) Create a **non-static inner class** inside the **DemoClass**. Move the **calculateArea**() function inside the **inner class**. Now call the **calcualteArea**() function from the **main**() function.

Question 2 (5) [CO2]

Remember how you raced with your friends in your childhood? Someone shouted: 3…2…1…GO! And you started running! Now, create a simple GUI application that shows 3…2…1…GO!
The GUI will have only 2 components in a Frame: A **Label** and a **Button**. The label will **not show any text** in the beginning and the **frame's layout** will be set to **FlowLayout**. When you press the button the first time, the label will show "**3…**". The next time you press the button, the label will show "**2…**", then "**1…**" and lastly "**GO!**".
Some parts of the code are done for you, you will need to complete the rest (Consider appropriate classes are imported).

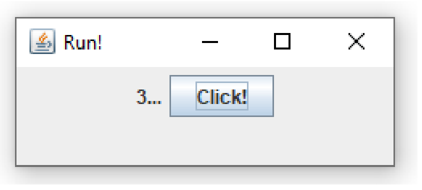
```
class Main {
    public static void main(String[] args) {
        JFrame fr = new JFrame("Run!");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.setSize(250, 100);

        JLabel label = new JLabel();
        JButton button = new JButton("Click!");

        // Write your code here.

        fr.setVisible(true);
    }
}
```

The GUI after you clicked the button once:



[Question 3 has 2 parts. Answer any one.] **Question 3 (2 + 3) [CO2]**

Consider the following code:

```
class Student{
    String name;
    double height;
    Student(String name, double height){
        this.name = name;
        this.height = height;
    }
}

public class ArrayListDemo{
    public static void main(String []args){
        // your code here
    }
}
```

a) Create an Arraylist of **Student** class inside the main function. Insert **five** Student objects into the Arraylist. Initialize the Student objects with random but proper values.

b) Add necessary code to sort the Arraylist created in (a) according to **height** in **descending order**.

OR

Question 3 (5) [CO2]

Create a thread class named **SumThread** that takes three parameters in the constructor: an **array**, an index **left**, another index **right**. The thread computes the sum of the passed array from index l to r **inclusive when it runs**. Consider the following main method of the Main class provided below and the array a. *You can assume the input n, will be multiple of 3*.

```
public class Main {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter num of elements: ");
        int n = scanner.nextInt();

        int[] a = new int[n];
        int total_sum = 0;

        for (int i = 0; i < a.length; i++) {
            a[i] = scanner.nextInt();
        }

        // Write your codes here
        // ....................

        System.out.println("Total Sum: " + total_sum);
    }
}
```

Now complete the following tasks:

a) Create the **SumThread** class by **implementing the Runnable interface** and define the constructor and other required methods. Pass the values of *left* & *right* properly to every thread constructor so that every object can compute the sum of **one-third non-overlapping** partitions.

b) Start three threads of the **SumThread** object. **The threads must run and compute the sum concurrently**.

c) After that, set the **accurate total sum from these three threads** into the ***total_sum*** variable in the main method of the Main class.
You must handle relevant exceptions. You don't need to rewrite the whole Main class. Just write the added codes.

Question 4 (5) [CO2]

You have received a classified file which contains many lines of data. Each line of the file contains two integers, separated by a single "**-**" (hyphen). Not all the lines are important in that file. If **one of the integers** in a line contains **four or more digits**, then that line is important. Your task is to **find out the important lines** in the file and **write** those lines in a **separate output file**. Check the example below for clarity:

Input File	**Output File**
101-199	1010-230
1010-230	1-29999
1-29999	5000-99999
20-30	….
5000-99999	
….	

Question 5 (5) [CO1]

Consider the class, **CartManager** given below. You will need to add a method: **applyPromo** in this class. The method takes a promoCode string as parameter and throws **InvalidPromoCodeException** based on following **two** criterias:

- Current **valid** promo codes are "**ENJOY50**" and "**HELLO100**". If any other promo code is passed as parameter in the applyPromo method, an **InvalidPromoCodeException** is thrown. The wrong promo code is passed to the constructor of **InvalidPromoCodeException** and the exception message is set as follows:

  ***HELLO200** is not a valid promo code right now.*

  Here **HELLO200** is the wrong promo code.

- For "**ENJOY50**" promo code, the total order price must be equal to or greater than **250**. For "**HELLO100**" promo code, the total order price must be equal to or greater than **500**. If these conditions are not met, then an **InvalidPromoCodeException** is thrown. The **promoCode** and the **totalPrice** is passed to the constructor of **InvalidPromoCodeException** and the exception message is set as follows:

  ***ENJOY50** cannot be applied for order price of **200**.*

  Here the promoCode is **ENJOY50** and the total price of order was **200**. Here's another example:

  ***HELLO100** cannot be applied for order price of **499**.*

If no exception was thrown, the **applyPromo** method should print:

Promo code applied successfully!

Note: Write only the codes which are not provided in the question. A method is provided in the **CartManager** class for calculating the total price.

```java
class CartItem {
    String name;
    double price;

    CartItem(String name, double price) {
        this.name = name;
        this.price = price;
    }
}
```

```java
class CartManager {
    ArrayList<CartItem> items;

    CartManager() {
        items = new ArrayList<>();
    }

    public void addItem(CartItem item) {
        items.add(item);
    }

    public double getTotalPrice(){
        double sum = 0;
        for (CartItem item : items)
            sum += item.price;
        return sum;
    }
}
```